

Form Tip

Letting Access dial the number

Several readers have requested an article about dialing a phone number from an Access form. These people want a Rolodex-style form, such as the one shown in Figure A, that lets them use the computer's modem to select a phone number from their client or customer data and then dial the number. In this article, we'll show you how to build such a form. We'll

with DOS commands. In the first part of this article, we'll explain the DOS commands you can use to dial a number, and we'll show you a batch file that automates the task. Then, we'll return to the Access environment and build the form that will use your batch file.

About your modem

Let's start with a general discussion of modems and how you operate them. A modem is a device that lets your computer communicate with the outside world. Although we aren't doing so in this article, you typically use a modem to communicate with other computers that are running electronic bulletin boards or online services such as CompuServe, Prodigy, or America Online. Once you install a modem, you operate it by using telecommunications software. Such software packages allow you to dial numbers and then interact with the other computer. They also help you transfer files between computers.

Sound complicated? Well, don't worry. You don't need to have any experience with telecommunications to make the modem dial a number. All you need to do is make sure the modem is installed properly in one of your computer's serial ports (and, of course, make sure you plug the modem's phone cord into a phone jack).

If you set up everything properly, simple DOS commands will dial the phone number. The commands route the modem commands to the modem's serial port. The modem will then pick up the commands and dial the number. After the modem dials the number and you hear the phone ringing, you pick up the receiver and wait for your client to answer.

also discuss the hardware you'll need to implement the technique and some problems that can arise.

An overview

As you'll see, the easiest way to control a modem is

Figure A

This form lets you dial a phone number with the click of a button.

IN THIS ISSUE

- Letting Access dial the number 1
- Creating a .PIF file for the batch files you run with the Shell() function 4
- Tips for formulating the phone number you send to the modem 6
- Creating custom menu bars for the first time 7
- Five pointers for creating custom menu bars 9
- Printing memo entries with the Can Shrink and Can Grow properties 11
- Using report parameters when the report isn't based on a table or query 15
- A better way to tie query criteria to a form control 16

What can go wrong?

There's one complication you may encounter. By telling the modem to dial the number, you give the modem control of the phone call. Certain modems won't return control to you when you pick up the phone.

This problem stems from the fact that the modem expects to hear carrier tones sent by the computer that answers the phone. Those carrier tones tell the modem what to do, and when the modem doesn't hear the tones, it gets confused. How it behaves in this state of confusion varies from modem to modem.

If you're lucky, your modem will do nothing. You'll be able to pick up the phone at your leisure and have your conversation. In this idealized situation, the modem will politely take itself out of the conversation some time after you begin talking.

Unfortunately, some modems won't do this. In fact, some types of modems will simply hang up when they determine the phone call won't be between two computers. Incidentally, the modems that have this problem are usually the more expensive ones. These modems are sophisticated enough to distinguish the human voice from the tones another computer's modem answers with.

If you own a modem that won't gracefully bow out of your phone call, you

must use a second DOS command to explicitly hang up the modem after you pick up the phone. The phone will then retain the connection.

Dialing a number with the ECHO command

We'll show you the batch file that controls the modem shortly. Let's first discuss the DOS command you'll use to send the modem commands to the serial port.

If you've written many batch files, you're probably very familiar with DOS' ECHO command. You may be surprised to hear that you use the ECHO command to dial a phone number. You've probably used ECHO only to display data onscreen. For example, when you issue the command

ECHO The Cobb Group

from the DOS prompt, the message *The Cobb Group* will appear on the next line.

However, ECHO isn't limited to sending messages to the screen. You can use the redirection operator (>) to send data to peripherals you install in the computer's parallel and serial ports. For instance, the command

ECHO The Cobb Group > COM1

Inside MICROSOFT ACCESS™

Inside Microsoft Access (ISSN 1067-8204) is published monthly by The Cobb Group.

Prices
Domestic \$59/yr. (\$7.00 each)
Outside US \$79/yr. (\$8.50 each)

Phone
Toll free (800) 223-8720
Local (502) 491-1900
Customer Relations Fax (502) 491-8050
Editorial Department Fax (502) 491-4200

Address

You may address tips, special requests, and other correspondence to

The Editor, *Inside Microsoft Access*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Advertising

For information about advertising in Cobb Group journals, contact Tracee Bell Troutt at (800) 223-8720, ext. 430.

Postmaster

Second class postage is pending in Louisville, KY. Send address changes to

Inside Microsoft Access
P.O. Box 35160
Louisville, KY 40232

Authorized Canada Post International Publications Mail (Canadian Distribution)
Sales Agreement #XXXXXX CANADA GST #123669673. Send returns to Canadian
Direct Mailing Sys. Ltd., 920 Mercer Street, Windsor, Ontario, N9A 7C2. Printed in
the USA.

Copyright

Copyright © 1994, The Cobb Group. All rights reserved. *Inside Microsoft Access* is an independently produced publication of The Cobb Group. The Cobb Group reserves the right, with respect to submissions, to revise, republish, and authorize its readers to use the tips submitted for both personal and commercial use.

The Cobb Group, its logo, and the Satisfaction Guaranteed statement and seal are registered trademarks of The Cobb Group. *Inside Microsoft Access* is a trademark of The Cobb Group. Paradox is a registered trademark of Borland International. dBASE III and dBASE III PLUS are registered trademarks of Ashton-Tate, a Borland International company. Microsoft, MS-DOS, and Access are registered trademarks of Microsoft Corporation. Microsoft Windows and Word for Windows are trademarks of Microsoft Corporation.

Staff

Editor-in-Chief	David Brown
Editing	Meredith Little
	Polly Blakemore
	Elizabeth Welch
Production Artist	Maureen Spencer
	Marguerite Stith
Design	Karl Feige
Managing Editor	Elayne Noltemeyer
Circulation Manager	Brent Shean
Editorial Director	Jeff Yocom
Publishers	Mark Crane
	Jon Pyles

Back Issues

To order back issues, call Customer Relations at (800) 223-8720. Back issues cost \$7 each, \$8.50 outside the US. You can pay with MasterCard, VISA, Discover, or American Express, or we can bill you.

will send the text *The Cobb Group* to the COM1 serial port. Of course, the modem you installed at this port won't know what to do with the string; nevertheless, the modem will receive the message.

To dial a phone number with a modem, you send a message the modem will recognize as the dial command. Assuming your modem adheres to the Hayes-modem standard (which almost all modems do), you use the command ATDT to dial the number. The *AT* portion means *attention* and, as you'd expect, gets the modem's attention. The *DT* portion means *dial tone* and opens the line for dialing a phone number. You follow the ATDT command with the phone number. For example, to call The Cobb Group's Customer Relations department by using your computer's modem, you'd issue the command

```
ECHO ATDT1-800-223-8720 > COM1
```

In order to hang up the modem, you use the ECHO command to send a different Hayes-modem command to the serial port—ATH. Again, the *AT* means *attention*, and the *H* portion means *hang up*. Putting it all together, you hang up the modem by using the DOS command

```
ECHO ATH > COM1
```

Creating the batch file

Now let's put these commands to work in a DOS batch file. We'll assume your modem has problems removing itself from your phone call, so we'll want to build in the hang-up feature.

To create the batch file, run your favorite text editor and type the commands. If you're working in the Windows environment, launch the Notepad application by double-clicking the Notepad icon (📝) in the Accessories group. When the application appears, you'll be ready to create a new file. Enter the commands

```
@ECHO OFF
ECHO Dialing number %1.
ECHO After you hear the phone ringing,
ECHO pick up the phone and
ECHO ATDT%1 > COM1:
PAUSE
ECHO ATH > COM1:
```

Then, use the File menu's Save As... command to save the file as *DIALNUM.BAT*. In the Save As dialog box, be sure to select the directory in which you store your Access database. For instance, if you store your database in the C:\ACCESS directory, type C:\ACCESS\DIALNUM.BAT in the Save As dialog box and click OK.

As you can see, the batch file contains several ECHO statements. The first, @ECHO OFF, tells DOS not to display the DOS prompt as it executes the batch file's commands. The next three ECHO commands simply display messages onscreen that tell you what's happening during the batch file's execution.

The last two ECHO commands do the real work. The first of them dials the number, and the second one sends the hang-up command, terminating the modem's role in the phone call. Between those ECHO commands sits the PAUSE command, which suspends the batch file until you press a key.

Why do you insert a PAUSE statement? Well, you don't want the modem to hang up until it has finished dialing the number and the phone is, in fact, ringing. This process takes time. Remember, the ECHO command doesn't actually dial the number—it only routes the modem command to the serial port. The modem executes the dial command while the batch file continues to the next command. As a result, the batch file will be ready to execute the next command long before the modem actually dials the number. The PAUSE command lets you decide when to resume the batch file and, therefore, send the hang-up command to the modem.

We also need to explain the %1 that appears in the batch file. The %1 represents the first parameter you include on the batch file's command line. By using this symbol, you can provide the phone number as an argument. For example, to dial The Cobb Group's number from the DOS prompt, you'd run the batch file followed by

```
DIALNUM 1-800-223-8720
```

When you hear the phone ring, you pick up the phone receiver and press a key on the keyboard. The batch file will then send

the hang-up command to the modem, and you can talk normally.

Running the batch file from Access with the Shell() function

Now that we've shown you how to run the batch file from the DOS prompt, let's bring the batch file to the Access environment. Let's start by discussing how you actually run the batch file from Access.

You can run any program by using the Access Basic function Shell(). The Shell() function accepts two arguments—the filename and a window style code that

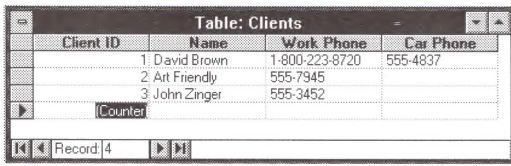
tells the function *how* it should run the file in the Windows environment.

However, in our example, we don't need to worry about the second argument. When you run a DOS executable such as a batch file, the Shell() command will run the program in full-screen mode. There's no sense in choosing a window mode when the batch file doesn't run in a window! If you'd like to take advantage of the possible window modes, you must create a .PIF file for the batch file. For more information, see "Creating a .PIF File for the Batch Files You Run with the Shell() Function."

Creating the form

We're now ready to create the Client table and Rolodex-style form. We'll use a simple Client table that contains a Counter field called Client ID as the primary key, a

Figure B



Client ID	Name	Work Phone	Car Phone
1	David Brown	1-800-223-8720	555-4837
2	Art Friendly	555-7945	
3	John Zinger	555-3452	

We'll create our sample form for this table.

Creating a .PIF file for the batch files you run with the Shell() function

In the accompanying article, we use the Shell() function to run a .BAT, or *batch*, file from an Access form. In that article, we explain that the Shell() function's second argument, which lets you choose a window style, is pretty much useless when you run a batch file. Shell() automatically runs batch files and other DOS programs in full-screen mode. Therefore, the second argument doesn't have an effect, because the program doesn't actually run in a window.

To make a batch file run in a window, you must create a .PIF file and run the .PIF file rather than the .BAT file. As you may know, .PIF files let you customize the DOS environment Windows creates to run the program. In this article, we'll show you how to create a .PIF file that will run the batch file DIALNUM.BAT in

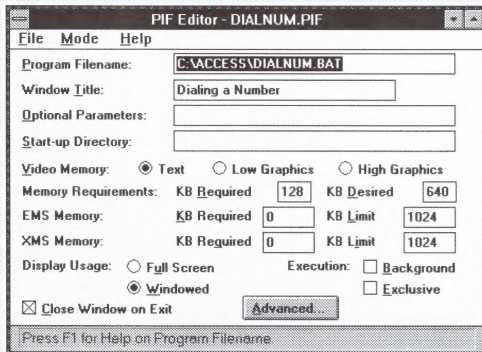
a window. We'll also describe the window-style options the Shell() function's second argument gives you.

Creating the .PIF file

To create a .PIF file, you first launch the PIF Editor application from Windows. You'll find the PIF Editor icon () in the Main program group. In an empty PIF form, you type the batch file's name in the Program Filename text box. Then, you enter the caption for the window in the Window Title text box. To create a window in which to run the batch file, you move to the Display Usage option group in the lower-left corner and then click the Windowed option.

That's all it takes. Figure A shows how the PIF form should look. Save the file by pulling down the File menu and clicking the Save As... command. Then, enter \ACCESS\DIALNUM.PIF in the Save As dialog box and click OK. If you store your Access database file in a different directory, use that directory name. Finally, close the PIF application and return to Access.

Figure A



You create this .PIF file to run the DIALNUM.BAT batch file in a window.

Name field, and two phone number fields—Work Phone and Car Phone. Figure B shows the table's data sheet with a few sample client records.

We'll use the Single-Column form wizard to create the basic form and then enter Design View to place the buttons and assign the Shell() function calls to the buttons' On Push properties. Start by highlighting the Client table in the Database window and clicking the New Form button (📄) on the tool bar. Next, click the Form Wizards button in the New Form dialog box. Then, in the following dialog box, select the Single-Column wizard and click OK. In the wizard's first dialog box, click the Fast Forward button (➤) to generate the default form. In the wizard's last dialog box, click the Design button. When you do, the wizard will generate the form shown in Figure C.

Figure C

We'll use the default single-column form in our example.

Next, place the Dial buttons on the form. First, open the tool box by pulling down the View menu and clicking the Toolbox entry. Then, select the Command Button

The Shell() function's window-style argument

Once you return to Access, you're ready to replace the DIALNUM.BAT batch file in the Shell() function with DIALNUM.PIF. Now that the batch file will run in a window, you can set the Shell() function's window style with the second argument. Table A lists the possibilities.

Now let's use the .PIF file in the accompanying article's Rolodex-style Client form. Enter Design View for the form, select the Dial button next to the Work Phone text box, and open the property sheet. In the On Push property, replace the existing Shell() function call with

```
Shell("C:\ACCESS\DIALNUM.PIF
➔ ![[Work Phone]]",2)
```

For our purposes, we'll run the batch file with the default window style, 2, which runs the file in a minimized window that retains focus.

Note: You must select a window-style code that lets the batch file retain focus. If you don't, you won't be able to interact

with the batch file. However, you may opt for a normal, minimized, or maximized window that retains focus. By doing so, you'll ensure that your Access form remains visible while the batch file dials the phone number.

Table A

Code	Window Style	Description
1, 5, 9	Normal with focus	Creates an ordinary window for the file and makes that window current while running the program
2	Minimized with focus	Creates a minimized window for the file but keeps the window current so that it receives your keystrokes
3	Maximized with focus	Creates a maximized window in which to run the file
4, 8	Normal without focus	Creates an ordinary window for the file but leaves the Access window current while running the file
6, 7	Minimized without focus	Creates a minimized window for the file but leaves the Access window current while running the file

tool (☐) and place a button to the right of the Work Phone text box. By default, the button will be quite large. You'll need to reduce the button to the size of the text box control. Next, open the property sheet by clicking the Properties button (☐) on the tool bar. Replace the Caption property's default entry with the label *Dial*. Then, assign the function call

```
=Shell("C:\ACCESS\DIALNUM |[Work Phone]|")
```

to the On Push property.

To create the other button, use the Edit menu's Duplicate command. This command will create a button that's the same size as the first button and has the Caption and On Push property values we just set. Position the new button just to the right of the Car Phone text box control. Then, move to the On Push property in

the property sheet and modify the function call so it reads

```
=Shell("C:\ACCESS\DIALNUM |[Car Phone]|")
```

You're now ready to use the form. Click the Form View button (☐) on the tool bar. Then, try clicking one of the Dial buttons. A friendly Cobb Group customer service representative will answer.

Notes

The Shell() function calls we use in our example grab the phone number directly from the form control. If you design your form to do the same, you must take care to store the phone numbers exactly as you want to dial them. For instance, you must begin long-distance phone numbers with the digit 1. ♦

Tips for formulating the phone number you send to the modem

In the accompanying article, we showed you how to dial a phone number with your computer's modem. However, we didn't describe how the modem interprets the digit in the phone number you send. In this short article, we'll discuss several things you'll need to keep in mind as you formulate the phone number you send to the modem.

Punctuation in phone numbers

The first item we'll discuss is the punctuation you use in phone numbers. You may have wondered what the modem does with the hyphens we embedded in the numbers in the accompanying article. Well, the modem ignores them. They're simply optional. You can leave them in if it's more convenient, or you can strip them out. Furthermore, modems will ignore parentheses, so you can send a phone number in the format (800) 232-8720.

The comma character

One character that modems treat specially is the comma, which tells the modem to pause for about a second. (You can change the pause duration with certain modem commands; however, we won't discuss those advanced modem commands in this article.) Why would you ever want to pause while dialing a number? Well, the comma is almost always necessary when you need to dial out of your company's phone system by first pressing the 9 button. If you must dial 9 to get an out-

side phone line, you must send the number in the format 9,1-800-223-8720.

Protecting against Call Waiting

The last issue we'll discuss doesn't have a direct impact on the modem when you're dialing a number in order to reach a customer. It's more important when you're using your modem to communicate with another computer.

If you use a modem from a phone line that has Call Waiting, you may want to disable the service. As you may know, Call Waiting prevents you from missing calls when you're already on the phone. If another person calls while you're talking to someone, you hear a beep. You can then switch to the other caller and decide who you really want to talk to.

The problem is, the beep you hear breaks the connection between the modems. Fortunately, you can disable Call Waiting by dialing the four-digit code 1170. After you dial the four digits, the dial tone will return. You can then dial the number normally. When you want to disable Call Waiting for a phone call through the modem, you send the 1170 code and then a comma in order to wait for the dial tone.

Note that some areas don't have the ability to turn off Call Waiting. If, after dialing 1170, you hear a fast busy signal, you can't disable Call Waiting.

Creating custom menu bars for the first time

If you've worked with Access for long, you know how easy it is to customize a form to suit your particular needs. As you become more sophisticated in designing forms, you'll use more sophisticated tools. In this article, we'll show you a form-design tool that many beginning Access users are reluctant to take advantage of: We'll show you how to create custom menu bars for your forms.

The basics

Many new users are a little intimidated by creating custom menu bars. However, when you do it for the first time, you'll see that it's actually quite simple, and you'll probably want to create custom menu bars for many of your forms.

The basic steps for creating a custom menu bar are

- Planning your menu bar on paper (mapping out the commands and operations you want to include on the pull-down menus)
- Creating a macro group for each pulldown menu you want to define (since macros in the groups implement the individual options on the pulldown menu)
- Defining a macro that creates the menu bar by executing an AddMenu action for each pulldown menu
- Opening in Design View the form for which you're designing the menu and assigning the menu bar macro to the form's On Menu property

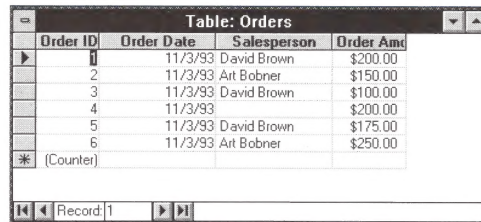
After you complete these steps, your new menu will appear in place of the standard Access menu when you open the form.

The example

To show you this procedure, we'll create a simple menu step by step. Let's start by

defining the table we'll use in our example. Figure A shows an Orders table, which contains a counter field named Order ID as the table's key field and additional fields named Order Date, Salesperson, and Order Amount.

Figure A



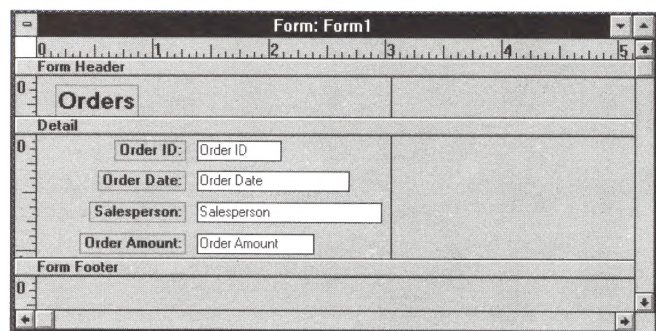
Order ID	Order Date	Salesperson	Order Amt
1	11/3/93	David Brown	\$200.00
2	11/3/93	Art Bobner	\$150.00
3	11/3/93	David Brown	\$100.00
4	11/3/93		\$200.00
5	11/3/93	David Brown	\$175.00
6	11/3/93	Art Bobner	\$250.00

We'll use this Orders table in our example.

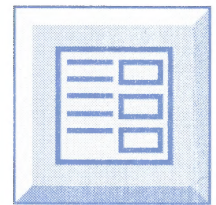
Let's next create the form for which we'll define the custom menu bar. Highlight the Orders table in the Database window and click the New Form button () on the tool bar. Then, click the Form-Wizards button in the New Form dialog box. When the next dialog box appears, select the Single-Column form wizard and click OK. In the Single-Column wizard's first dialog box, click the Fast Forward button (). When the wizard's final dialog box appears, click the Design button. Access will generate the form shown in Figure B.

You don't need to make any additional changes to the default form at this point. For now, save the form by pulling down the File menu and selecting the Save As... option. Enter *Orders Browse* in the Save As dialog box and click OK.

Figure B



We'll design a custom menu for this simple form.



Form Tip

Building the custom menu bar

We'll next turn to the custom menu bar you want to build for the Orders Browse form. The initial step is to plan the menu system. In this simple example, you'll create two pulldown menus. The first, called Form, contains two options—Print and Close. The second pulldown menu, called Records, will offer five options—Find, for searching the table for certain data, and the four standard navigational actions, Go To

Table A

The MenuOrdersForm Macro		
Macro Name	Action	Action Arguments
Print	DoMenuItem	Menu Bar = Form Menu Name = File Command = Print
Close	Close	

Table B

The MenuOrdersRecords Macro		
Macro Name	Action	Action Arguments
Find	DoMenuItem	Menu Bar = Form Menu Name = Edit Command = Find
Go To Previous	GoToRecord	Record = Previous
Go To Next	GoToRecord	Record = Next
Go To Top	GoToRecord	Record = First
Go To Bottom	GoToRecord	Record = Last

Previous, Go To Next, Go To Top, and Go To Bottom. In a real-world situation, you'd probably want to create more menu options than this. For our purposes, we want to keep things simple.

Once you've planned your menu, you create macro groups for the two pulldown menus. Table A shows the macro group named MenuOrdersForm, which defines the macros for the form's pulldown menu. Table B shows the macro group named MenuOrdersRecords, which defines the Records menu options.

To create these macros, click the Macro button in the Database window and then click the New button. When the new Macro window appears, click the Macro Name button (📄) on the tool bar and enter the macro names, actions, and action arguments. Save the macros by using the Save As... command on the File menu.

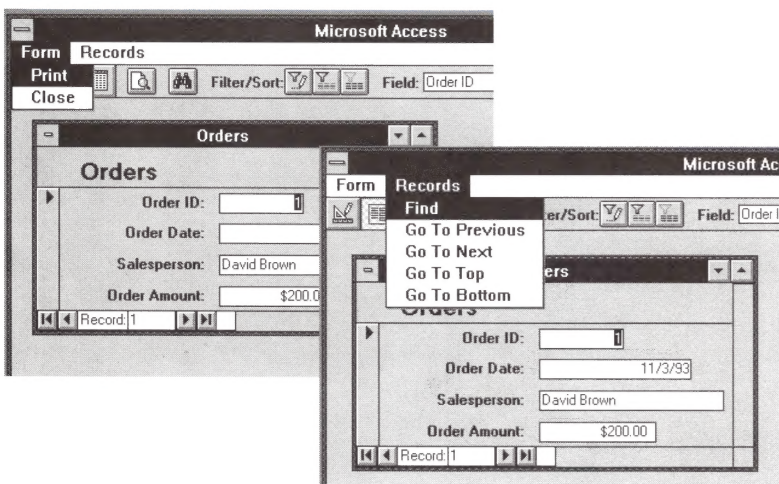
Next, you create another macro that defines the menu bar. This macro is simply a collection of AddMenu items that identify the menu bar's pulldown menus. To create this macro, make sure the Database window lists the macros and click the New button. Then, when the new Macro window appears, enter the actions and action arguments listed in Table C. When you've finished, pull down the File menu and click the Save As... command. Type the macro name *MenuOrdersBar* and click OK.

You now assign the menu macro to the Orders Browse form's On Menu property. Return to the form and enter Design View. Then open the property sheet by clicking the Properties button (🔧) on the tool bar. In the On Menu property, click the drop-down arrow and then click MenuOrdersBar in the selection list. By doing so, you tell the form to display the MenuOrdersBar macro's menu system while you view the form. Figure C shows the two menus.

Suggestions for your first custom menu bar

After you've worked through the example we've presented here, you'll probably want to create menu bars for your own forms. We have one piece of advice: Don't underestimate the importance of planning your menu system. You should choose a form you're very familiar with and write

Figure C



These screen shots show how the two pulldown menus look when you use the Orders Browse form.

down the commands and operations you routinely need while you use that form. Since you're so familiar with the form, the task of arranging those commands and operations will come naturally.

Conclusion

In this article, we showed you how to create a custom menu bar for the Orders Browse form, and we walked through the basic steps necessary to put a useful menu on the screen. There's plenty more you can do with menus in Access. If you're inter-

Table C

The MenuOrdersBar Macro	
Action	Action Arguments
AddMenu	Menu Name = Form Menu Macro Name = MenuOrdersForm
AddMenu	Menu Name = Records Menu Macro Name = MenuOrdersRecords

ested in enhancing the appearance of your menus, read "Five Pointers for Creating Custom Menu Bars," below. ❖

Five pointers for creating custom menu bars

In "Creating Custom Menu Bars for the First Time," on page 7, we walked you through the basic steps you use to create a custom menu bar for a form. In this article, we'll show you five design tips that will bring a professional touch to your menus.

1. Duplicating Access menu items

We've already used our first tip to create the example we showed you in the previous article. We include it here in the list only because it's so important for you to know that you can duplicate Access menu items in your custom menus by using the DoMenuItem action.

The situation we described in the previous article contains two examples. You used the DoMenuItem action in the macro MenuOrdersForm to implement the Form menu's Print option. You used the action in the macro MenuOrdersRecords to create the Records menu's Find option. Figure A shows the Print dialog box that appears when you select the Print menu option on your custom menu. As you can see, it's the same dialog box that Access produces when you issue Access' Print... command on the File menu.

One additional tip we'll point out here is to include an ellipsis (...) with names of menu options that lead to dialog boxes.

You've probably noticed that menu option names in all Windows-based applications end with the ellipsis when selecting an option name calls a dialog box.

2. Creating access keys for the menu options

In previous articles, we've told you how to create access keys for command buttons and controls on forms. Well, you can create access keys for menu options as well.

Let's first review the technique. As you may remember, you typically define an access key by inserting an ampersand (&) just before the letter you want to make the access key. When you view the form, the access key letter will be underlined in the button name or control label.

When you define an access key for a menu item, you do the same thing.

You insert an ampersand in the name of the macro that defines the menu item. To do this, you must open in Design View the macro group that defines the option's pulldown menu. You can then go down the list of

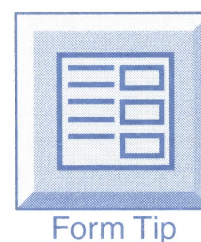
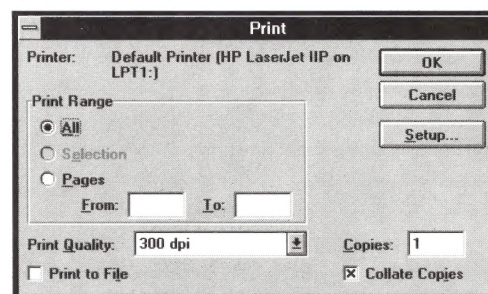


Figure A

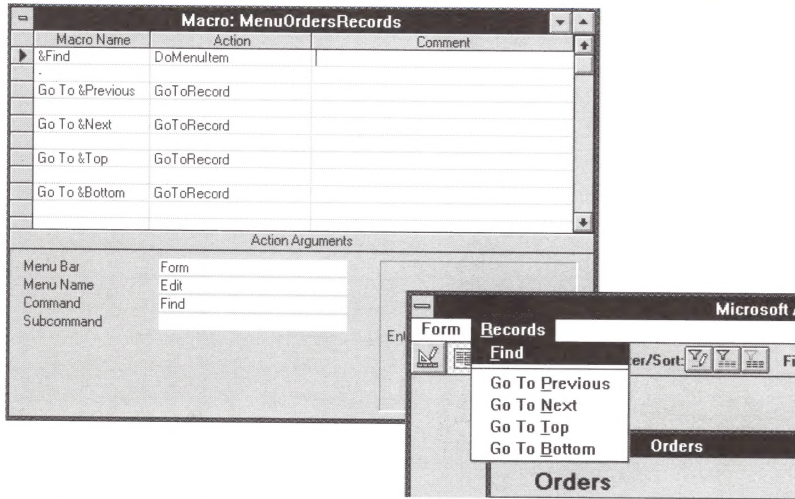


The DoMenuItem action you use in your custom menu to print the form produces the same Print dialog box that Access provides.

macro names, defining access keys for all the menu options in the macro group.

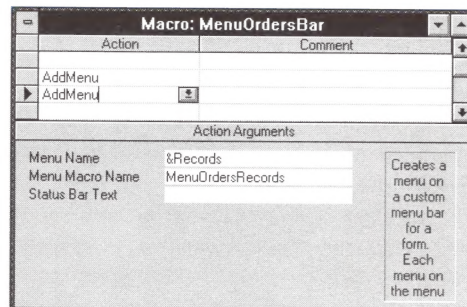
Figure B shows the access keys you can define for the options on the Records

Figure B



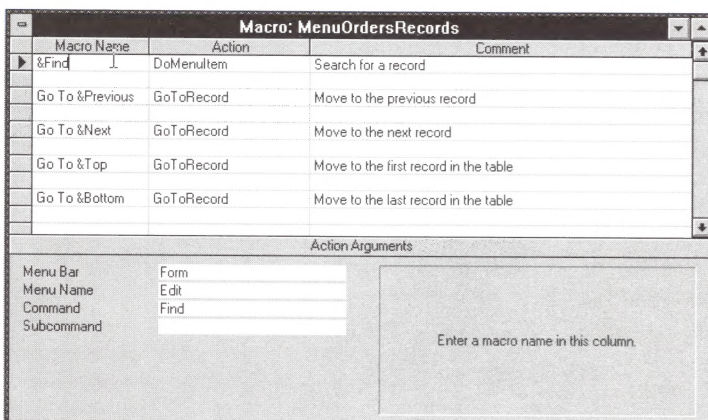
Here you can see the ampersands that define the access keys and how the access keys appear in the pull-down menu.

Figure C



You define an access key for the pull-down menu in the AddMenu action's Menu Name argument.

Figure D



When you highlight a menu option, Access displays the text you enter in the option's Comment cell in the Status Message area at the bottom of the screen.

menu. The figure shows the macro group and the resulting pull-down menu.

You can also define access keys for the menu items on the menu bar. To do so, you open in Design View the macro that assembles the menu bar. You then insert the ampersand in the Menu Name action argument of the AddMenu action.

Returning to our example, suppose you want to make *R* the Records menu's access key. Figure C shows the ampersand you insert in front of the *R* in the Menu Name argument's *Records* entry.

3. Providing captions for the menu items

We'll next turn to providing captions for your menu items. You can define captions for both the pull-down menu names on the menu bar and the individual items on the pull-down menus. Access displays both types of captions in the status bar when you highlight the menu item.

At the time you modified the macro MenuOrdersBars in the previous section, you may have noticed the Status Bar Text argument of the AddMenu action. You enter in this action argument the caption you want for the menu bar item.

How you provide captions for the menu options on the pull-down menus isn't obvious. You enter captions for the menu options in the Comment column of the macro groups. Just type the caption in the first line of the menu option's macro. Access will display your entry in a macro's Comment cell when you highlight the menu option. Figure D shows the captions we entered for the menu items in the Records menu.

4. Inserting separator bars to group related commands

Separator bars are another feature you'll want to include in your custom menus. You insert separator bars in your pull-down menus to cluster the menu's related commands. By grouping related commands, you make the menu easier to read. When you pull down the menu, you'll be able to find the entry you need, since you can skip sections of options that obviously don't apply.

You create separator bars in the macro groups. To do so, you first create a row

between the macros of the two options you want to separate and then type a hyphen in the new row's Macro Name cell.

To illustrate this technique, we'll show you how to separate the Find option from the four Go To options in the example's Records menu. First, you'll need to open the MenuOrdersRecords menu in Design View. Next, insert a row after the Find macro and enter a hyphen in the Macro Name column. Figure E shows the result.

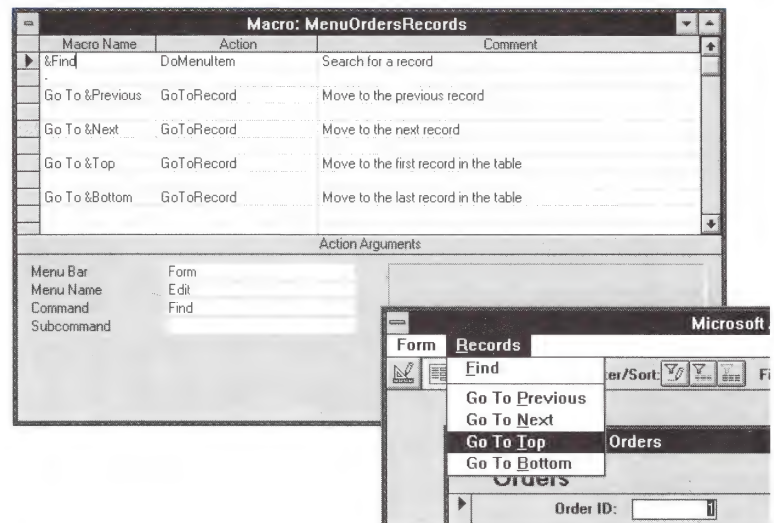
5. Turning off the menu

As our final tip, we'll tell you how to turn off the menu bar. You open the form in Design View, display the property sheet by clicking the Properties button (🔧) on the tool bar, and assign =0 to the On Menu property. When you then click the Form View button (📄) on the tool bar, the menu bar will no longer appear at the top of the Access window.

Conclusion

In this article, we embellished "Creating Custom Menu Bars for the First Time"

Figure E



You can insert a separator bar to distance the Find option from the four Go To menu options.

with five design tips. These tips will help you give your menus the professional look commercial Windows applications have. ❖

Printing memo entries with the Can Shrink and Can Grow properties

If you use Memo fields in your tables, you may experience a few problems when you try to print your memo entries on a report. In this article, we'll show you everything you need to know about printing long text or memo entries on a report. We'll first describe how the Can Grow and Can Shrink properties help you. We'll then discuss the Keep Together property.

Understanding the Can Shrink and Can Grow properties

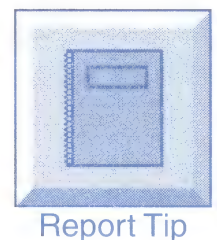
By setting a text box control's Can Shrink and Can Grow properties, you tell Access whether it can resize the control according to the entry's size. When you set Can Shrink to Yes, Access will shrink the text box if the entry is shorter than the control.

When you set Can Grow to Yes, Access will lengthen the text box if the memo entry is longer than the control. By assigning No to those properties, you tell Access to print the control as it appears in the report's Design View.

The Can Grow property is the more important of the two when you're printing memo entries. It lets you design a coherent layout for the report section but lets Access expand the control when the text box contains a long entry.

An example

Let's build a sample report that uses the Can Grow property to print a memo entry. Table A on page 12 shows the structure of the Help Info table, which stores information about various Access features. The Help Info table uses the counter field Topic ID for its key and uses the text field Topic



Report Tip

Description to store the topic's title or description. The third field, Help Text, is a memo field that stores the information the Access Help system provides on the topic. We'll create a report that prints the memo entries in the Help Text field.

After you create the table, you must load it with data. Let's walk through the steps for using the Help system to enter our test data. You'll first start the Help system and then look up the topics you want to include. When you find the topics, you'll copy the Help text to the Clipboard and then paste it into the Help Text field in the Help Info table.

Start by pulling down the Help menu and selecting the Search... option. Then, begin typing the first entry. After you type *Can*, you'll see the *CanGrow* and *CanShrink* entries appear at the bottom of the selection list. Double-click the *CanGrow* entry to see the topic descriptions for that keyword. The Help entry *CanGrow, CanShrink Properties* will appear. Click the Go To button to see the text for that Help topic.

When the Help window appears, you must copy the text to the Clipboard. Do so by pulling down the Edit menu and

selecting Copy. The Help system will open the Copy dialog box. Before you can copy the text, you must select it. First, click the scroll bar thumb and drag it to the bottom of the scroll bar. Then, shift-click below the last line of the Help entry. Since the cursor resided at the very top of the entry, shift-clicking at the bottom will select the entire text. Next, click the Copy button.

Finally, return to the Help Info table in Access and enter the topic title—*CanGrow, CanShrink Properties*—in the Topic Description field. Next, move to the Help Text field and paste in the Help text by using the Edit menu's Paste command. You'll repeat this process for the Keep Together property, the Dim statement, and the Global statement.

To create the report, highlight the Help Info table in the Database window and click the New Report button (📄) on the tool bar. In the New Report dialog box, click the Blank Report button. When the new Report window appears, click the Field List button (📋) on the tool bar. Then, drag the three fields from the field list to the report's Detail section. Next, rearrange and format the text box controls as shown in Figure A.

Note that we've boldfaced the label controls for the Topic ID and Topic Description text boxes. You create this effect by clicking on the label so that you select the label control rather than the text box. You then click the Bold button (B) on the tool bar. We also resized the Help Text text box and deleted its label.

Figure B shows how this first try at the report looks. Notice that you can see only the first couple of lines of the Help Text memo entry. Also, you can see only the first line of the Topic Description entry. You can't see the full entries because the Can Grow properties of those text boxes are set to *No*.

To see the full entries in the Topic Description and Help Text fields, you set the Can Grow properties of the text box controls. Select the Topic Description field's text box and click the Properties button (🔧) on the tool bar. Then, change the Can Grow property to *Yes*, as shown in Figure C.

Table A

The Help Info Table			
Key	Field Name	Data Type	Field Properties
🔑	Topic ID	Counter	
	Topic Description	Text	Field Size = 50
	Help Text	Memo	

Figure A

After you place the fields, your report should look like this.

Figure B

Topic ID	1	CanGrow, CanShrink Properties
Topic Description	Can Grow, Can Shrink	See Also: Access Basic
Topic ID	2	KeepTogether Property
Topic Description	Keep Together prop	See Also: Access Basic
Topic ID	3	Dim Statement
Topic Description	Dim statement	See Also: Example
Topic ID	4	Global Statement
Topic Description	Global statement	See Also: Example

Before you modify the Can Grow properties, the report prints only the portion of the entry that will fit in the control.

Figure D shows the first three pages of the resulting report. As you can see, the full memo entries print. You may also notice that each memo begins on a new page. We'll have more to say about that later.

The Can Grow and Can Shrink properties of report sections

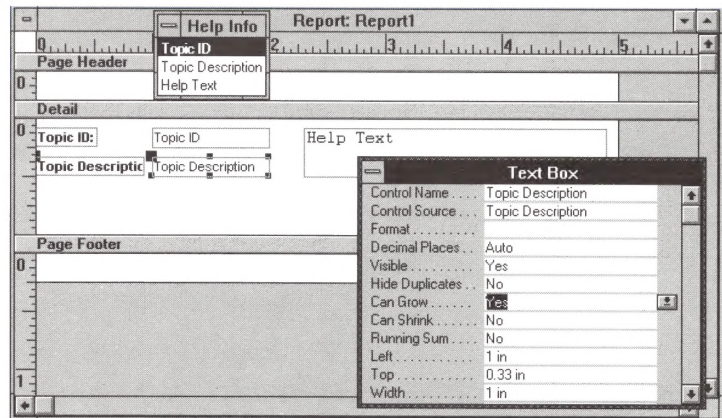
So far, the report formats the long text and memo entries with only the text box controls' Can Grow property. However, report sections also have Can Grow and Can Shrink properties. As you'd expect, by setting these properties, you tell Access whether it can resize the report section based on the size of the controls in the section.

As you can see in Figure D, the Detail section that contains the memo control grew with the large memo entries. Consequently, you might think the Detail section's Can Grow property has the default setting *Yes*. Actually, the de-

fault setting is *No*—just as it is for a text box control. Access automatically changes it when you assign *Yes* to the Can Grow property of a control the section contains. In the above example, the instant you change the Topic Description control's Can Grow property to *Yes*, Access changes the Detail section's Can Grow property to *Yes*.

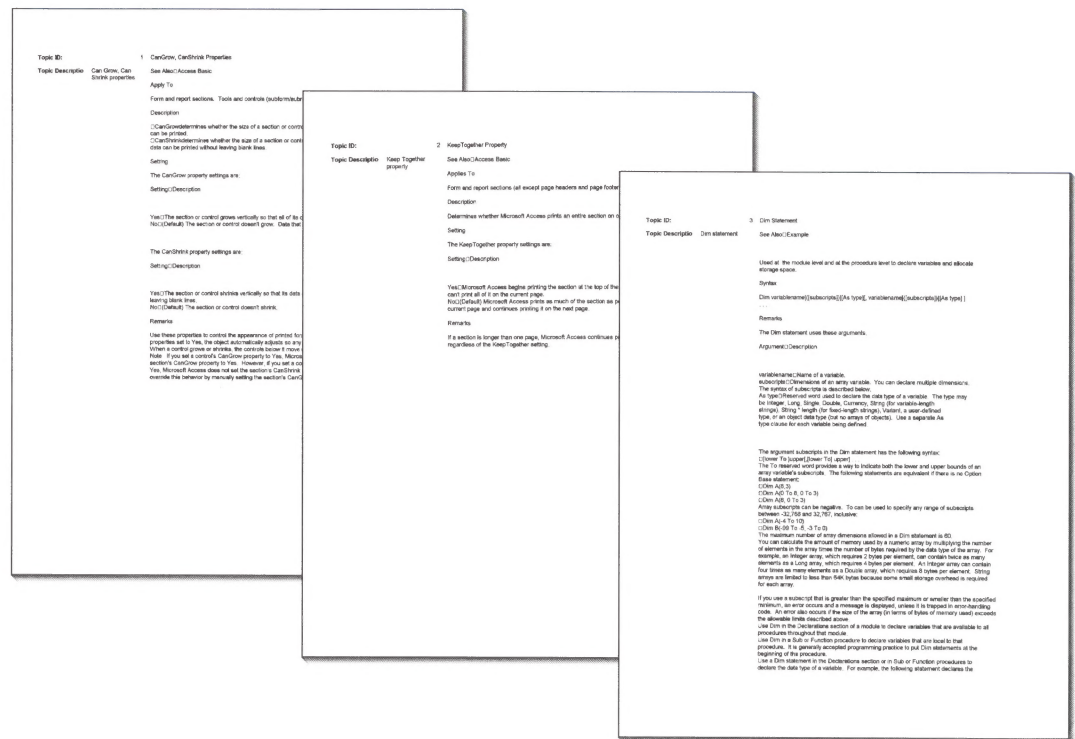
Access automatically sets the section's Can Grow property because the vast

Figure C



In the property sheet, you set the controls' Can Grow properties to *Yes*.

Figure D



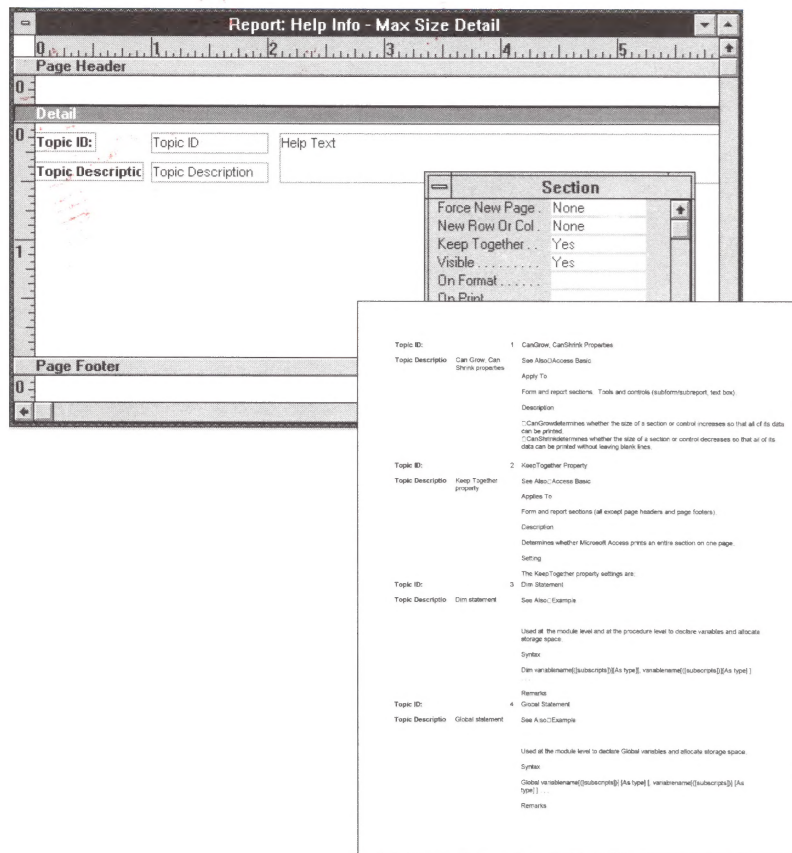
After you set the control's Can Grow properties to *Yes*, you can see the full text and memo entries.

majority of the times you set a control's Can Grow property to *Yes*, you want the control to print the entire entry. If the section's Can Grow property remained *No*, the control could grow only as much as the space allotted to the section.

However, you sometimes want to limit the size of a section when you want to let the section's controls grow. For instance, you may wish to define a maximum size for a section. In this case, you'd set the section's Can Grow property to *No*. That way, the section won't grow beyond the maximum amount of space, even if you assign *Yes* to the Can Grow properties of the section's controls.

Suppose you want to confine the Help Text memo entries to a maximum of two inches. You'd drag the lower boundary of the Detail section to the two-inch mark on the vertical ruler. You'd then set the Detail section's Can Grow property to *No*. Figure E shows the elongated Detail section and the report that results.

Figure E



You can set the Detail section's Can Grow property to *No* to define a maximum size for the memo field's text box.

Using the Keep Together property

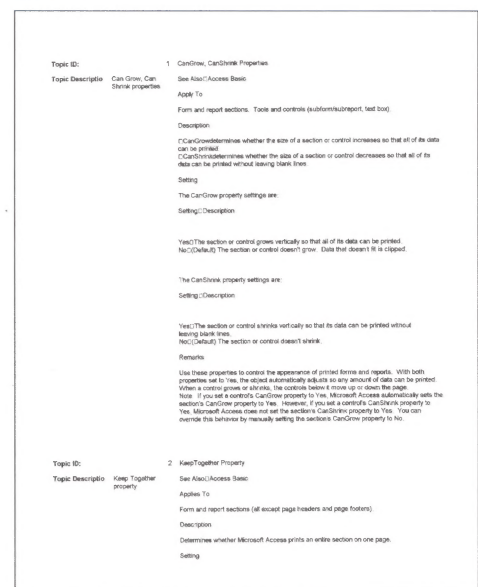
As we pointed out, each memo entry in the report begins at the top of a new page. You may be wondering why. After all, there's plenty of room at the bottom of most pages to fit a few lines of the next entry. Well, the report starts the memo entries on separate pages because the entries are too big for two to fit on a page. By default, the report will move to the next page when it can't lay out all the section's controls on the current page.

However, you can choose whether you want the report to allow a section to span two pages. If you select the section and open the property sheet, you'll find a property named *Keep Together*. By default, the property will be *Yes*. If you change it to *No*, the report will follow each section with the next, regardless of where on the page the section ends. To see how this works, change the *Keep Together* Detail section of our sample report. When you set the property to *No*, the report will print as shown in Figure F.

Notes

In our example, we created a blank report and placed the fields' text box controls one at a time. When we did, Access assigned the default value of *No* to the Can Shrink and Can Grow properties, and it assigned

Figure F



When you set a section's *Keep Together* property to *No*, the report won't begin a new page when the next section won't fit on the page.

Yes to the Keep Together property. If you create reports with the wizards, the initial property values may be different. The wizard makes the changes that are necessary for the style of report you choose.

Also, everything we've discussed about the Can Shrink, Can Grow, and Keep To-

gether properties in reports applies to the corresponding properties in forms. Of course, those commands affect only how Access prints the form. When you're viewing the form, the text box controls provide scroll bars to let you browse the full entry. The size of the text box remains constant. ❖

Using report parameters when the report isn't based on a table or query

In your May issue of *Inside Microsoft Access*, you printed a useful article entitled "Using Parameters in Reports to Provide Information as You Print." The article explains that you can use parameters in reports just as you can in queries. You simply enter a parameter in the Control Source property of a report's text box control. When you run the report, Access will prompt you for the parameter value just as it does when you run a parameter query. Access will then use the value you typed to print the text box control.

I've learned that the technique works only when the report is based on a table or query. I often design reports that exclusively use Domain functions, such as `DSum()` and `DLookup()`. Since I don't base these reports on a table or query, I quickly had trouble implementing your technique.

In my specific situation, I base the report's data on a field named GENID. The report prints data by using a certain range of the GENID field's entries, and I want the report's title to indicate this range. Because I can't use your technique, I currently enter Design View to adjust the report's title every time I print the report. What I need is

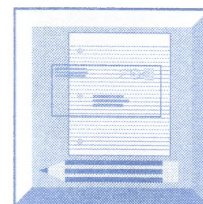
a way to prompt for the report title as your article does. Any ideas?

Joe Okoneski
West Linn, Oregon

Mr. Okoneski has found an interesting problem that has a fairly easy solution. While it's true you should base the report on a table or query before Access can prompt for parameters, you don't actually need to use the table or query in the report. In other words, you can base the report on a table or query but then place no field controls on the report. The table can just sit there in the background.

There's one complication. When you print the report, Access will print the Detail section for every record in the table. If you placed the `DLookup()` and `DSum()` functions in the Detail section, Access will print many copies of your report.

To ensure that Access will print just one copy, you should base the report on a table containing no data. We suggest you create a new dummy table that will always remain empty: Its sole purpose is to exist for the report.



Letters

Subscribe to Inside Microsoft Access Resource Disk! H

Do you wish that you could experiment with the forms, reports, tables, macros, modules, and queries we regularly feature in *Inside Microsoft Access* but don't have the time or patience to create them? If so, you may want to subscribe to Inside Microsoft Access Resource Disk. Once you subscribe, we'll send you a disk loaded with all the useful tips featured in that month's *Inside Microsoft Access*. A disk icon marks the included articles.

A six-month subscription to Inside Microsoft Access Resource Disk costs \$29. A full one-year subscription is \$49. If you don't want to subscribe but would like the forms, reports, tables, macros, modules, and queries in a particular issue of *Inside Microsoft Access*, you can purchase a single disk for only \$9.95.

To subscribe or order a specific month's disk, just call Customer Relations at (800) 223-8720. Outside the US, please call (502) 491-1900.

Microsoft Access
Technical Support
 (206) 635-7050



Please include account number from label with any correspondence.



A better way to tie query criteria to a form control

I want to thank you for a very informative article ("Using Query By Form to Replace a Parameter Query's Dialog Box") in your June issue of *Inside Microsoft Access*. In that article, you base the query criteria on the values in a form's controls.

As you noted, it's important to formulate the criteria in such a way that the query won't limit the records by a field entry when the form control doesn't contain a value. For instance, if you often want to limit the query results by a state code entry, you'd include the State field in the query. Then, in the Criteria cell, you'd enter an expression that refers to a field on the form in which you type the actual criteria. Now, suppose you don't always want to limit the query results by the State field's entries. When the form control is blank, you want the query to return records with any entry in the State field.

In your article, you accomplished this by placing the expression

```
Like IIf([Forms]!
➤ [Customer Names From A State]!
➤ [Input State] Is Null, "*",
➤ [Forms]! [Customer Names From A State]!
➤ [Input State])
```

in the State field's Criteria cell. This complex IIf() function returns an asterisk if the form control is blank, and the function returns the form control's value if the control is not blank. When the Input State control is

blank, the query uses the criterion *Like "*" ,* which doesn't limit the query results.

I have a suggestion for simplifying and possibly improving your expression. I suggest you use

```
[Forms]! [Customer Names From A State]!
➤ [Input State] & "*"
```

Notice I don't use an IIf() function. If the form control doesn't contain a value, the expression evaluates to only an asterisk, which doesn't limit the query results. On the other hand, if the form control *does* contain a value, the query criterion limits the query to records that begin with the form control's value.

Although my expression does the same thing as yours, it has one important difference: It doesn't limit the records to exact values. In many situations, this change is an improvement. For instance, if you use my type of expression to limit the query results by a Last Name field, you can enter *A* in the form control and the query will return the people whose last names begin with *A*.

Thaddeus M. Sendzimir
 Waterbury, Connecticut

We thank Mr. Sendzimir for his tip. His expression is much simpler. For many types of queries, letting the query select records based on the partial entry is a real plus. However, you must always remember that the query isn't selecting records with exact matches. ♦

Can you export a query?

Many readers wrote us about a problem in the August issue's "A New Export Option in Access 1.1 Lets You Create Word for Windows Data Files." That article states that you have the option of selecting a table *or* a query for exporting to a Word for Windows data file. In reality, you can select only a table. Access doesn't list queries when you select the source of the export operation.

If you want to export the results of a query, you must convert the query to a make table query. After you run the make table query, you can export the new table. You can easily convert your existing select queries to a make table by issuing the Query menu's Make Table... command.

Note that you don't need to save the query as a make table query. You can run the query from the query's Design View and then close the query without saving your changes.

